

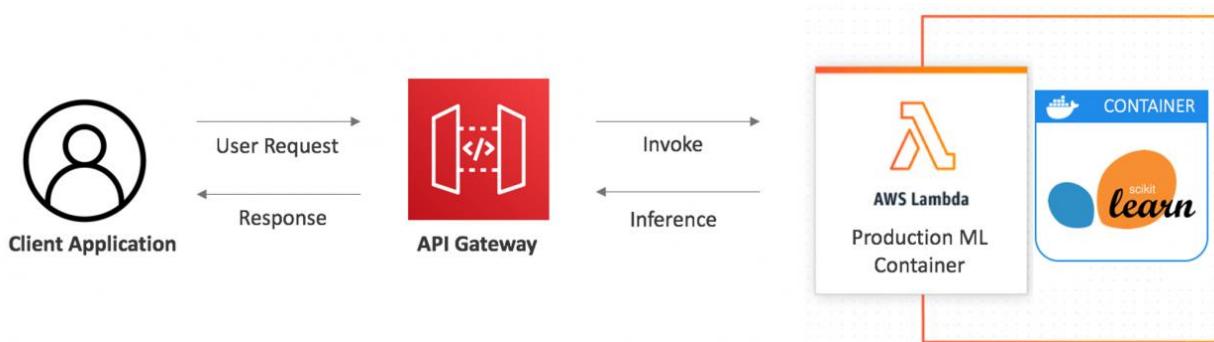
## Python/ Deep Learning Programming Final Project

Title: Serverless Deep Learning Framework

Team Name: Winter is Coming

Members: Srinivas Rahul Sapireddy (Individual Submission)

### Introduction



Here we introduce a clothes classification service. Where the user takes the picture of the clothes they want to wear. Then the classifier tells the user you are trying to buy in the clothes category.

Here I have used keras and tensorflow for image classification for classifying the pictures of different clothes.

Here I have used AWS Lambda for deploying the model. Lambda is a service from AWS which allows us to deploy many different machine learning models.

### Problem Statement

Here we send the picture with the URL to the model that we are going to deploy with AWS Lambda and then the service will reply with different classes and we will also have some score for it.

Here we use tensorflow-lite internally.

### Questions Answered?

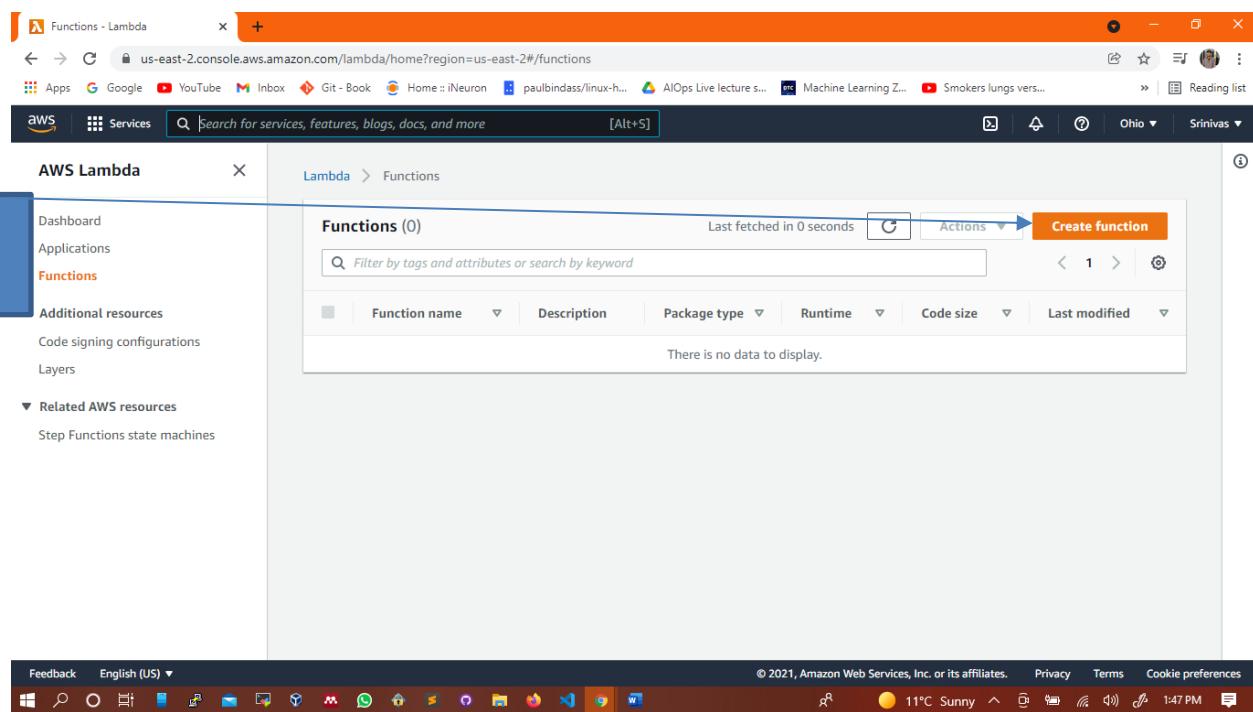
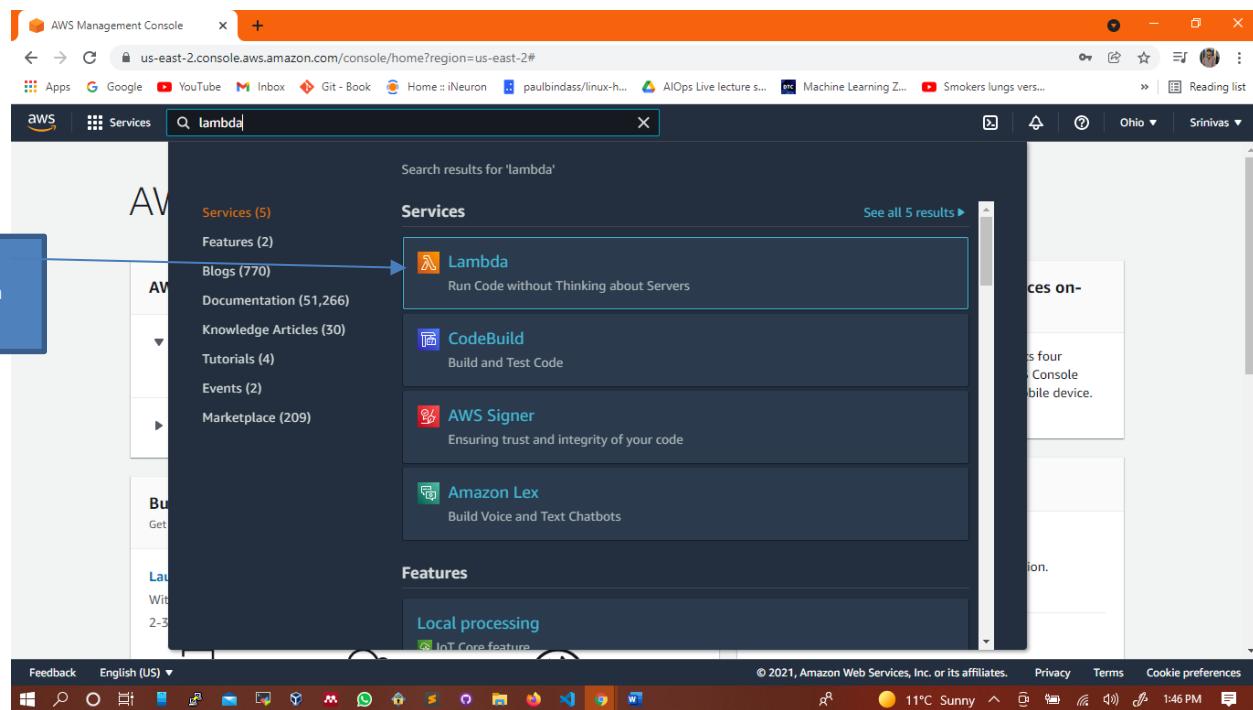
What AWS Lambda is and difference between other approaches?

What is the tensorflow-lite and why it is better for this particular use case?

We will package everything in to a docker container and deploy it to AWS Lambda. Finally, we will expose this Lambda function as web service API gateway.

**Serverless** is the concept of removing infrastructure considerations for deploying code. Instead of having to manage servers and infrastructure by ourselves, a serverless service takes care of that for us and only charges according to use.

## AWS LAMBDA



## Creating lambda function

The screenshot shows the 'Create function' wizard in the AWS Lambda console. A blue box highlights the 'Author from scratch' option, which is selected with a radio button. The other three options—'Use a blueprint', 'Container image', and 'Browse serverless app repository'—are shown with their respective descriptions. Below this, a section titled 'Basic information' contains fields for 'Function name' (set to 'myFunctionName') and 'Runtime' (set to 'Info'). The runtime dropdown menu also lists 'Python 3.9'. The top navigation bar includes links for 'Lambda', 'Services', and a search bar.

The screenshot shows the 'Create function' wizard in the AWS Lambda console. A blue box highlights the 'Environment' section, which includes fields for 'Function name' (set to 'test'), 'Runtime' (set to 'Python 3.9'), and 'Architecture' (set to 'x86\_64'). Below these, there's a 'Permissions' section with a link to 'Change default execution role'. At the bottom, there's an 'Advanced settings' section with a link to 'Edit advanced settings'. The top navigation bar includes links for 'Lambda', 'Services', and a search bar.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with tabs for 'Services' and a search bar. Below the navigation bar, a green banner message says: 'Successfully created the function test. You can now change its code and configuration. To invoke your function with a test event, choose "Test".'. The main area is titled 'test' and contains a 'Function overview' section. It shows a thumbnail for the function, a 'Layers' section (0 layers), and buttons for '+ Add trigger' and '+ Add destination'. To the right, there's a 'Description' field (empty), 'Last modified' (3 seconds ago), and a 'Function ARN' field (arn:aws:lambda:us-east-2:813985511592:function:test). Below this are tabs for 'Code' (which is selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The bottom of the screen shows a Windows taskbar with various icons.

We can see the code of this function here.

The screenshot shows the AWS Lambda console with the 'Code source' tab selected. The interface includes a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is highlighted), 'Deploy', and 'Changes deployed'. On the left, there's a sidebar with an 'Environment' section and a file tree showing 'test - /' and 'lambda\_function.py'. The main area displays the code for 'lambda\_function.py':

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from Lambda!')
8     }
9
```

The bottom of the screen shows a Windows taskbar with various icons.

The screenshot shows the AWS Lambda console interface. A green success message at the top states: "Successfully created the function test. You can now change its code and configuration. To invoke your function with a test event, choose "Test".". The main area displays the code for a Lambda function named "lambda\_function" in a file called "lambda\_function.py". The code is as follows:

```
import json
def lambda_handler(event, context):
    # TODO Implement
    return "TEST"
```

Parameters in the Lambda function –

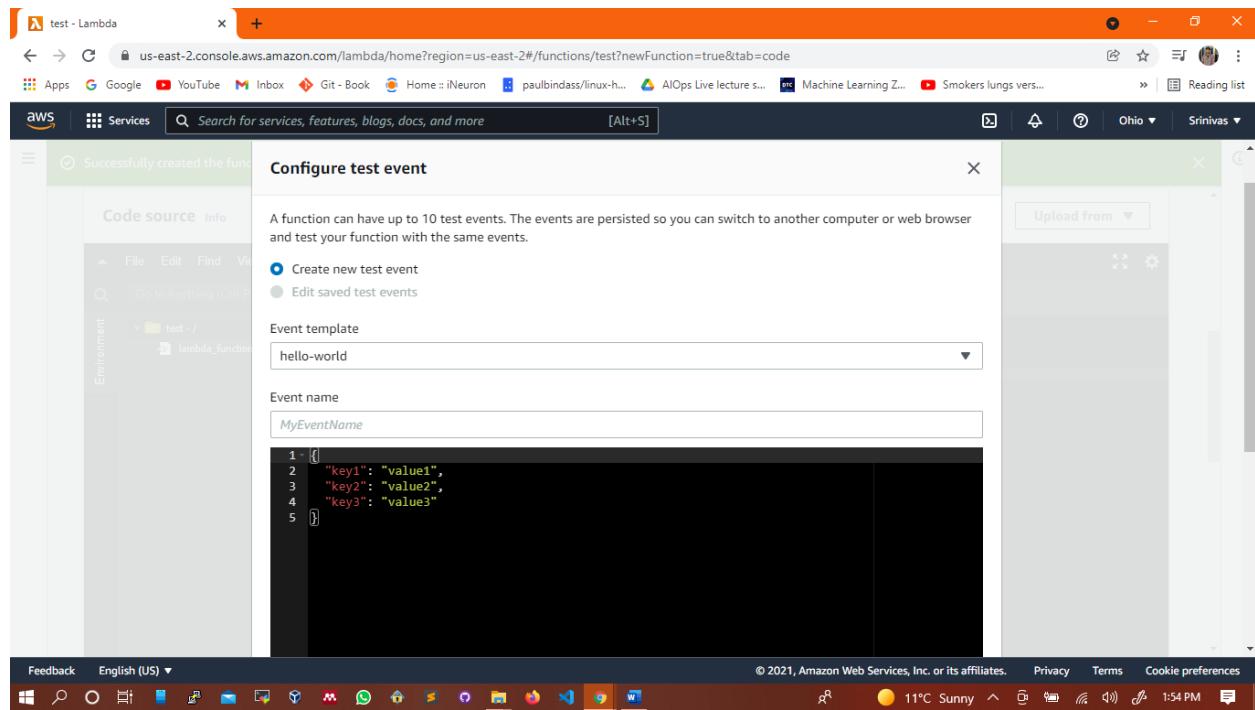
[1] Event – whatever we want to pass to the lambda function we pass it here.

[2] context – None

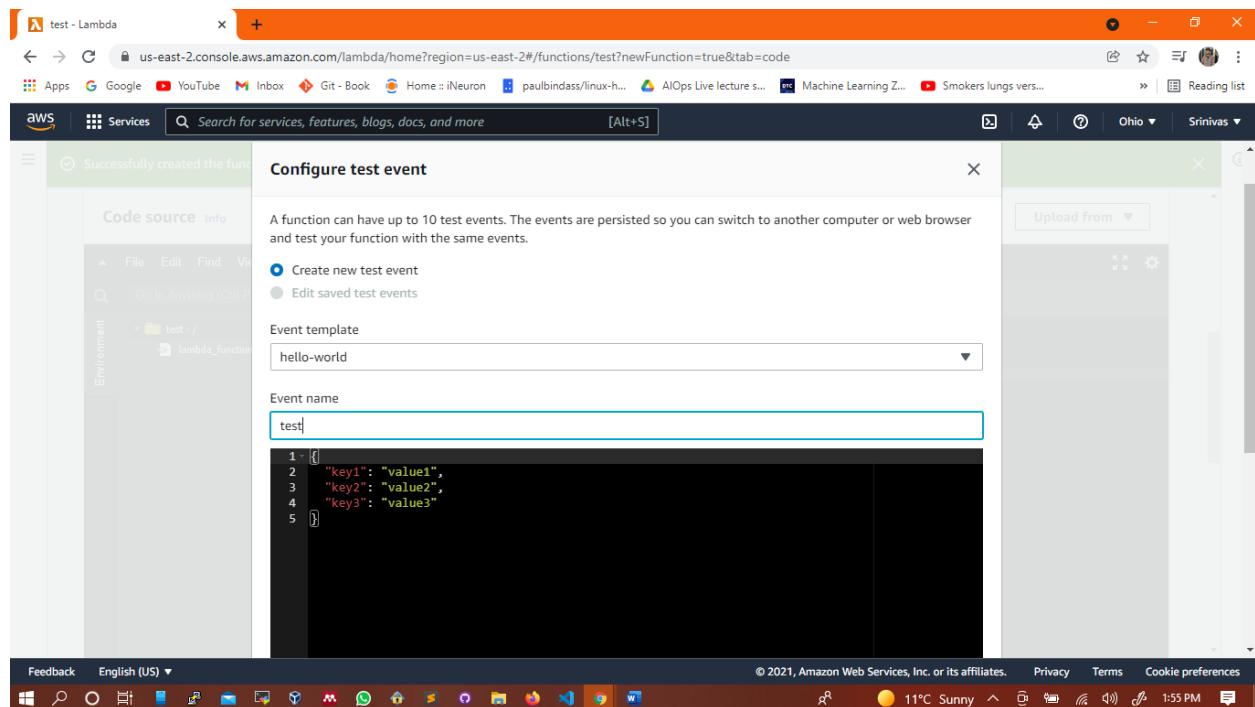
The screenshot shows the AWS Lambda code editor with the same basic structure as before. The code has been modified in the "lambda\_function.py" file to include a print statement:

```
import json
def lambda_handler(event, context):
    # TODO Implement
    print("Parameters: ", event)
    return "TEST"
```

Print the event parameter here.

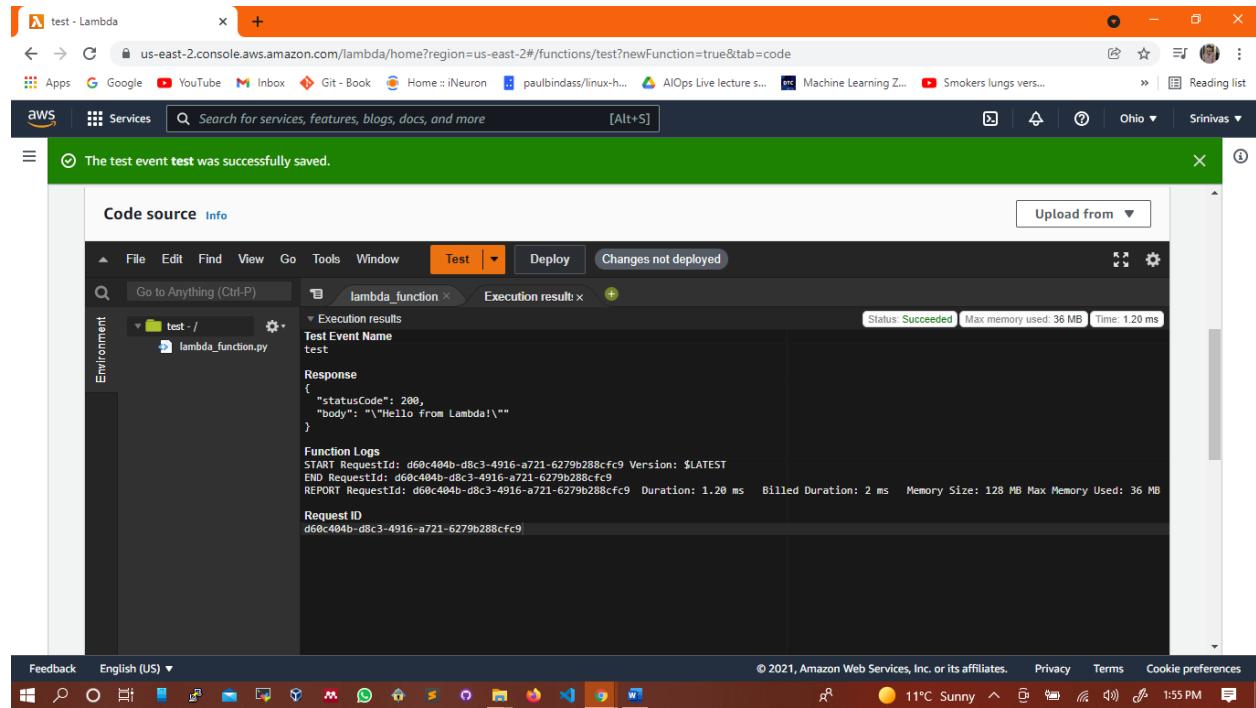


Here we get the test event. Whatever we want to test we pass it to the event parameter.

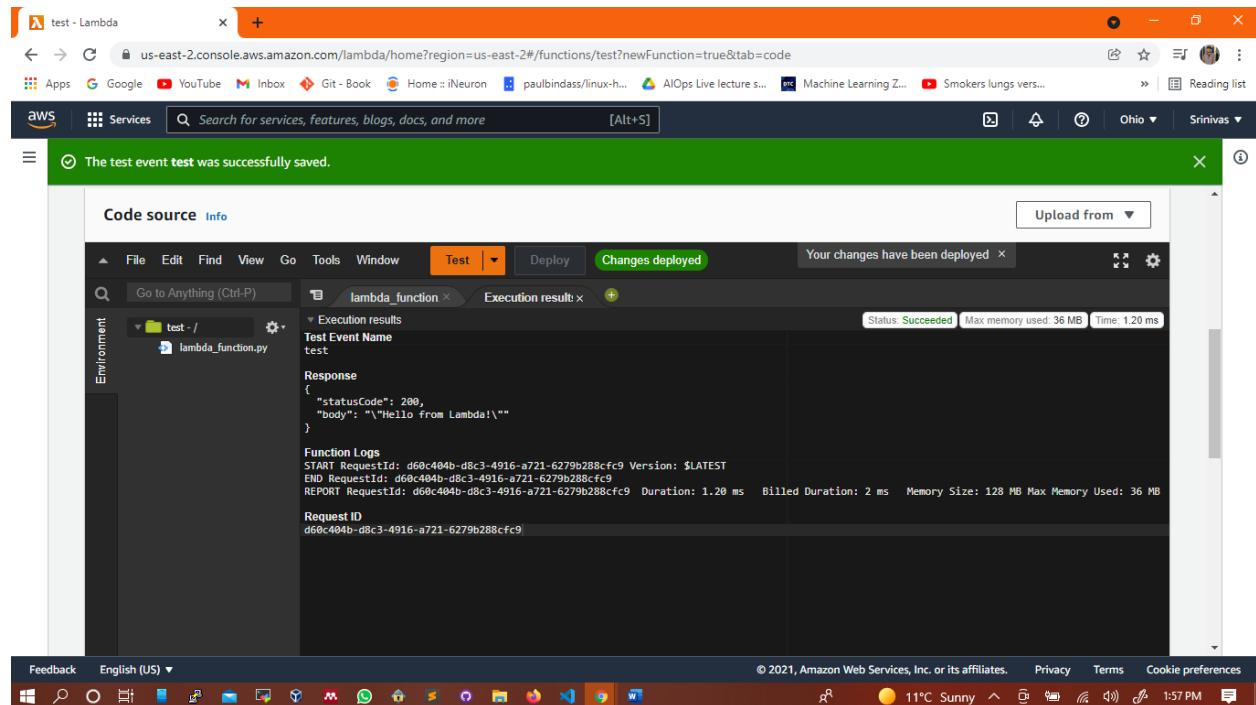


Creating the new test event.

## Testing lambda function



Changes are not deployed here we need to update the lambda function. Click on Deploy.



Then click Test

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with tabs for 'AWS' and 'Services'. A search bar says 'Search for services, features, blogs, docs, and more'. Below the search bar, a message says 'The test event test was successfully saved.' The main area is titled 'Code source' with an 'Info' tab. It has a toolbar with 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (which is currently selected), 'Deploy', and 'Changes deployed'. On the left, there's a sidebar for 'Environment' and a file tree showing 'test - /' and 'lambda\_function.py'. The central pane displays 'Execution results' for a test event named 'test'. It shows the response 'TEST' and function logs. The logs include details like START RequestId, END RequestId, Duration, Billed Duration, Memory Size, and Max Memory Used. The status is 'Succeeded' with a duration of 1.55 ms and memory usage of 36 MB. At the bottom, there's a footer with links for 'Feedback', 'English (US)', and various system icons.

Here we see the parameters we pass and the response.

Here we pass a URL parameter to the lambda function.

## ACCESSING THE URL

The screenshot shows the AWS Lambda console with a modal window titled 'Configure test event'. This window is used to define test events for the function. It has two options: 'Create new test event' (radio button) and 'Edit saved test events' (radio button, which is selected). Below this is a section for 'Saved Test Events' with a dropdown menu set to 'test'. Underneath is a code editor containing a JSON object:

```
1 - {
2   "url": "some-url"
3 }
```

The rest of the screen shows the same environment and file tree as the previous screenshot, with the message 'The test event test was successfully saved.' at the top. The footer is identical to the first screenshot.

The screenshot shows the AWS Lambda console interface. The top navigation bar includes links for Apps, Google, YouTube, Inbox, Git - Book, Home :: iNeuron, paulbindass/linux-h..., AIOps Live lecture s..., Machine Learning Z..., Smokers lungs vers..., Reading list, Ohio, and Srinivas. The main area displays a green success message: "The test event test was successfully saved." Below this, the "Code source" tab is selected in the toolbar, which also includes "Info", "Test", "Deploy", and "Changes not deployed". The code editor shows a single file named "lambda\_function.py" with the following content:

```
import json

def lambda_handler(event, context):
    # TODO implement
    url = event["url"]
    print("Parameters:", event)
    return {"Prediction": "Some_Prediction"}
```

The screenshot shows the AWS Lambda console interface, identical to the previous one but with different results. The top navigation bar and the green success message are the same. The "Code source" tab is still selected. The "Execution results" tab is now active in the toolbar, showing the status "Succeeded" and metrics: Max memory used: 36 MB and Time: 1.27 ms. The results pane displays the following information:

- Execution results
- Test Event Name: test
- Status: Succeeded
- Max memory used: 36 MB
- Time: 1.27 ms
- Response:

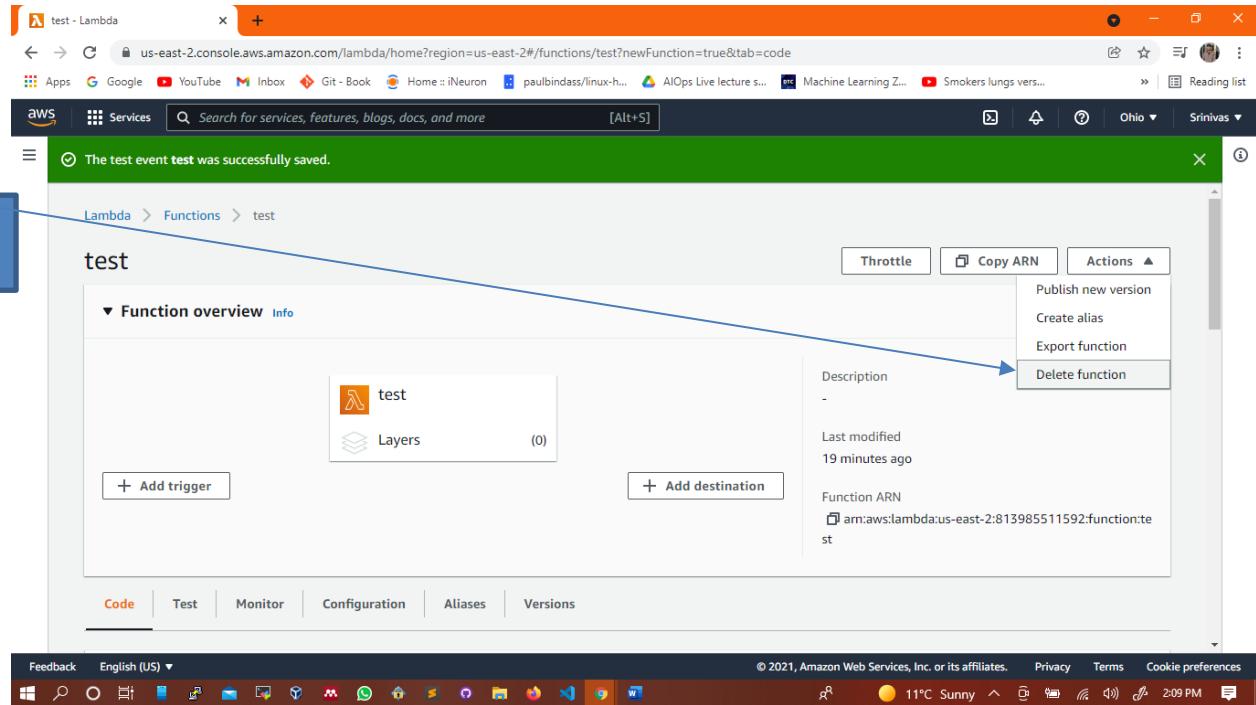
```
{"Prediction": "Some_Prediction"}
```
- Function Logs:

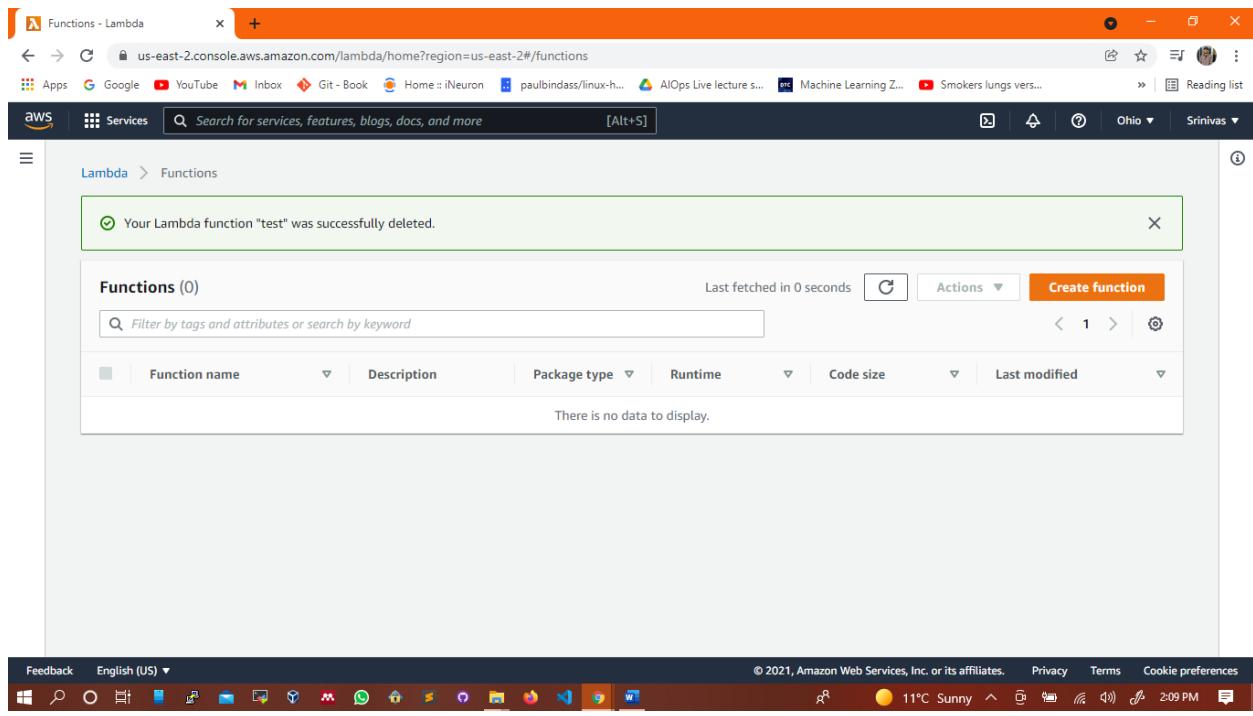
```
START RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82 Version: $LATEST
PARAMETERS: {"url": "some-url"}
END RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82
REPORT RequestId: 41fe7b38-e278-43b0-8ec6-0d123878bf82 Duration: 1.27 ms Billed Duration: 2 ms Memory Size: 128 MB Max Memory Used: 36 MB
```
- Request ID: 41fe7b38-e278-43b0-8ec6-0d123878bf82

## Advantages of using lambda function

There is no need to create any EC2 instances and no need to think about any servers. Here we just need to write a function and just user it.

No need to think about the infrastructure for serving the URL for serving the models. Here we pay only when lambda function is responding and don't pay when it is idle and not in use. Here it is serverless and we don't need to worry about servers.





Function is deleted.

## TENSORFLOW LITE

AWS Lambda has limit of the file size if can serve. That is less than 50MB zip file.

At present we have docker and have a larger limit. Up to 10GB. But if we have a larger image size it will be difficult to store the image. After invoking the Lambda function, it will take some time to initialize. Also initializing the tensorflow also takes lot of time. And it also takes more RAM in it.

Here we use a lighter version of tensorflow which is tensorflow-lite. This will only focus on inference. It is not useful for training the models but only used for inferencing the models.

## PREPARING CODE FOR THE LAMDA

### IMAGE – TESTING THE IMAGE

We prepare docker from base images prepared from AWS.

<https://gallery.ecr.aws/lambda/python>

The screenshot shows a web browser window with the URL [gallery.ecr.aws/lambda/python](https://gallery.ecr.aws/lambda/python). The page title is "ECR Public Gallery - AWS Lambda". The main content is about the "python" base image, which is described as "AWS Lambda base images for Python". It features the Lambda logo, a "Verified account" badge, and a download count of "2.6M+ Downloads". Below this, there are tabs for "About", "Usage", and "Image tags". The "About" tab is selected. The text on this tab explains that the images contain the Amazon Linux Base operating system, the runtime for a given language, dependencies, and the Lambda Runtime Interface Client (RIC), which implements the Lambda Runtime API. It also mentions that the Lambda Runtime Interface Client allows your runtime to receive requests from and send requests to the Lambda service. A note at the bottom indicates that more information about the composition of the base images can be found at <https://github.com/aws/aws-lambda-base-images>.

Docker build -t clothing-model .

wipty docker run -it --rm -p 8080:8080 clothing-model:latest

The screenshot shows a Visual Studio Code interface with the title "Dockerfile - chapter-08-serverless - Visual Studio Code". The left sidebar shows a file tree with files like chapter-08-model-test.ipynb, clothing-model-v4.tflite, convert.py, Dockerfile, lambda\_function.py, README.md, and test.py. The main editor area contains a Dockerfile:

```

FROM public.ecr.aws/lambda/python:3.8
RUN pip3 install --upgrade pip
RUN pip3 install keras_image_helper --no-cache-dir
RUN pip3 install https://raw.githubusercontent.com/alexeylegribov/serverless-deep-learning/master/tf/tflite_
COPY clothing-model-v4.tflite clothing-model-v4.tflite
COPY lambda_function.py lambda_function.py
CMD [ "lambda_function.lambda_handler" ]

```

The terminal tab at the bottom shows command-line output:

```

PS C:\Users\sapir\Desktop\Final Project\chapter-08-serverless> pip3 install requests
Collecting requests
  Using cached requests-2.26.0-py2.py3-none-any.whl (62 kB)
Requirement already satisfied: idna<4,>2.5 in c:\python310\lib\site-packages (from requests) (3.3)
Collecting certifi>=2017.4.17
  Using cached certifi-2021.10.8-py2.py3-none-any.whl (149 kB)
Requirement already satisfied: charset-normalizer<2.0.0 in c:\python310\lib\site-packages (from requests) (2.0.8)
Requirement already satisfied: urllib3<1.27,>1.21.1 in c:\python310\lib\site-packages (from requests) (1.26.7)
Installing collected packages: certifi, requests
Successfully installed certifi-2021.10.8 requests-2.26.0
WARNING: You are using pip version 21.2.3; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Python310\python.exe -m pip install --upgrade pip' command.
PS C:\Users\sapir\Desktop\Final Project\chapter-08-serverless> python test.py
{'dress': -1.8682897099177465, 'hat': -4.761244773864746, 'longsleeve': -2.3169829845428467, 'outwear': -1.0625699758529663, 'pants': 9.887154579162598, 'shirt': -2.8124303817749023, 'shoes': -3.66283130645752, 'shorts': 3.2003610134124756, 'skirt': -2.602339506149292, 't-shirt': -4.835044860839844}
PS C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>

```

```
PS C:\Windows\System32\cmd.exe - docker run -it --rm -p 8080:8080 clothing-model:latest
Server: Docker Engine - Community
Engine:
  Version:          20.10.10
  API version:     1.41 (minimum version 1.12)
  Go version:       go1.16.9
  Git commit:      e2f74ad
  Built:           Mon Oct 25 07:41:30 2021
  OS/Arch:         linux/amd64
  Experimental:    false
containerd:
  Version:          1.4.11
  GitCommit:        5b46e404f6b9f661a205e20d59c982d3634148f8
runc:
  Version:          1.0.2
  GitCommit:        v1.0.2-0-g52b36a2
docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>wlnpty docker run -it --rm -p 8080:8080 clothing-model:latest
'wlnpty' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>wlnpty docker run -it --rm -p 8080:8080 clothing-model:latest
'wlnpty' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker run -it --rm -p 8080:8080 clothing-model:latest
INFO[0000] exec '/var/runtime/bootstrap' (cwd=/var/task, handler=)
INFO[0152] Received signal                           signal=interrupt
INFO[0152] Shutting down...
WARN[0152] Reset initiated: SandboxTerminated

C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker run -it --rm -p 8080:8080 clothing-model:latest
INFO[0000] exec '/var/runtime/bootstrap' (cwd=/var/task, handler=)
INFO[0269] extensionsDisabledByLayer(/opt/disable-extensions-jwigqn8j) -> stat /opt/disable-extensions-jwigqn8j: no such file or directory
WARN[0269] Cannot list external agents               error="open /opt/extensions: no such file or directory"
START RequestId: 0a9df45-8d48-4c2a-8c95-9799f669908b Version: $LATEST
END RequestId: 0a9df45-8d48-4c2a-8c95-9799f669908b
REPORT RequestId: 0a9df45-8d48-4c2a-8c95-9799f669908b Init Duration: 1.21 ms Duration: 2461.93 ms    Billed Duration: 2462 ms      Memory Size: 3008 MB    Max Memo
ry Used: 3008 MB
```

## CREATING LAMBDA FUNCTION

Here we need to deploy the docker image to lambda. We publish the docker images in AWS ECR. ECR is where we publish the docker images.

### ECR – Elastic Container Repository

```
pip install awscli
```

```
aws ecr create-repository --repository-name clothing-tflite-images
```

```
Administrator Command Prompt - aws configure
Microsoft Windows [Version 10.0.19042.1348]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>aws configure
AWS Access Key ID [None]: AKIA33BKRHCUJ05CTACH
AWS Secret Access Key [None]: XyGcoJwODR7yZT1qTweiy8lCvxMr0c0C58cBNtR
Default region name [us-east-2]: us-east-2
Default output format [None]: json
```

```
Anaconda Prompt (anaconda)
(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws ecr create-repository --repository-name clothing-tflite-images
Provided region_name 'US_East' doesn't match a supported format.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [US_East]: us-east-2
Default output format [None]:

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws ecr create-repository --repository-name clothing-tflite-images
unable to locate credentials. You can configure credentials by running "aws configure".

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [us-east-2]:
Default output format [None]:

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws ecr create-repository --repository-name clothing-tflite-images
An error occurred (AccessDeniedException) when calling the CreateRepository operation: User: arn:aws:iam::813985511592:user:srinivas is not authorized to perform: ecr:CreateRepository on resource: arn:aws:ecr:us-east-2:813985511592:repository/clothing-tflite-images because no identity-based policy allows the ecr:CreateRepository action

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>aws ecr create-repository --repository-name clothing-tflite-images
{
  "repository": {
    "repositoryArn": "arn:aws:ecr:us-east-2:813985511592:repository/clothing-tflite-images",
    "registryId": "813985511592",
    "repositoryName": "clothing-tflite-images",
    "repositoryUri": "813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images",
    "createdAt": 1638230559.0,
    "imageTagMutability": "MUTABLE",
    "imageScanningConfiguration": {
      "scanOnPush": false
    },
    "encryptionConfiguration": {
      "encryptionType": "AES256"
    }
  }
}

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>
```

"repositoryUri": "813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images"

## AWS ECR Container Service

The screenshot shows the AWS ECR Container Service console. On the left, there's a sidebar with navigation links for Amazon ECS, Amazon EKS, and Amazon ECR. Under ECR, 'Repositories' is selected. The main area shows a 'Private repositories (1)' table. The table has columns: Repository name, URI, Created at, Tag immutability, Scan frequency, Encryption type, and Pull through cache. One row is listed: 'clothing-tflite-images' with URI '813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images', created on 'November 29, 2021, 18:02:39 (UTC-06)', tag immutability 'Disabled', scan frequency 'Manual', encryption type 'AES-256', and pull-through cache 'Inactive'. There are buttons for 'View push commands', 'Delete', 'Edit', and 'Create repository'.

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
clothing-tflite-images	813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images	November 29, 2021, 18:02:39 (UTC-06)	Disabled	Manual	AES-256	Inactive

Then we need to publish the image we have created.

We need to login to the registry to push there.

```
aws ecr get-login --no-include-email | sed 's/[0-9a-zA-Z=\]\{\!20,\}/PASSWORD /g'
```

The output of this will help to login to the registry and we will be able to push the registry.

```
$(aws ecr get-login --no-include-email)
```

```
[Anaconda Prompt (anaconda)]
Stored in directory: c:\users\saipr\appdata\local\pip\cache\wheels\4e\c9\5d\eb6899f8332d6ac88e1bd31c7fcbb6e28be8049c372297d8db
Successfully built sed
Installing collected packages: sed
Successfully installed sed-0.3.1
WARNING: You are using pip version 21.2.4; however, version 21.3.1 is available.
You should consider upgrading via the 'C:\Users\saipr\anaconda3\python.exe -m pip install --upgrade pip' command.

(base) C:\Users\saipr\Desktop\Final Project\chapter-08-serverless>aws ecr get-login --no-include-email | sed 's/[0-9a-zA-Z-]\{20,\}/PASSWORD/g'
$' is not recognized as an internal or external command,
operable program or batch file.

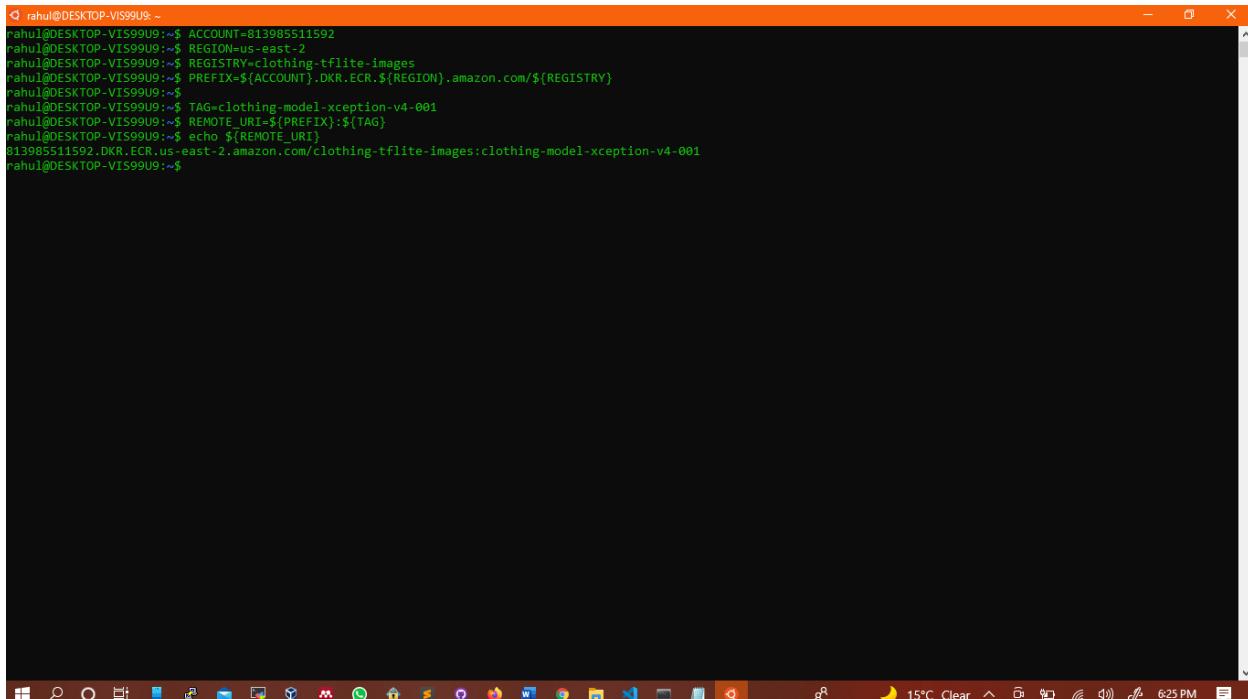
(base) C:\Users\saipr\Desktop\Final Project\chapter-08-serverless>aws ecr get-login --no-include-email
docker login -u AWS -p eyJwYXlsb2FkIjoiSURzQUVHTWxISng0NHdJNmhtCVgCwLdLSUxR0G1VmI1QyRnJBU2llicD1vekd20TBxRzRNMS9sZTixNTBmY0w5dG1zcXNQeEx20
5R8d591emg0Mzdt0Wx1ymEU2xJaT2FHJO2x0idhIpMy9jTHwaxBQ01xSwtDyKfVTNjsWd0tF4K02HeV1oYkr0UFBle1UrdndThQm1vB1ZE5tZQzNV2grm1iaT5sm2m7BqR0f5jBFmNDWFVEmphaj9FK
1pJ2ENrb1lMmfVNys3dGdXtnhUVK1zSwRb21VZQe1ZRT1dBNVJ0m2NzREVJ1UKRGy4Vge0edkYzhjcjd2WS9VOUROX16Smzpd2z14Lze1mNVTv14mPQVWhVpxRhoQz1NIAvkhZQm5V2B1n3pY081cGz1Vxd61cxzXnwV1InNfd2zuUm021uQ
3VYVmb0Su1YzJdeQsy1802X2m0EVx1d1H0nR02QzQs1pNkheQ8tYf1mzHNLw09hXfWmDF1zVnBrgt0a21DzQxbNvad1UtvexWhvSwSu1g2h2JchNv4BtRwnb1kbgHcm1c1N29vegVQz
zNeXg5yZhsca80e1m0kp2m1iaGovcTfG01psR0J0NEcncmc4Eue0H0dnd0BRkQTBl2zz22uDNbWQj01sZ1NvFf1R0d0tMbymhMeElyUmxtEz50Vh1y3R2u3m0u10251QxVcn1k0Q1nMrjt3t3hVgxleOpUd
1pblASQ0M4V4HpGmz20XBV8x1m5akm0131zVmlxUn0dtYzBxD0ZnRE1k0Wn1Qf1mzHNLw09hXfWmDF1zVnBrgt0a21DzQxbNvad1UtvexWhvSwSu1g2h2JchNv4BtRwnb1kbgHcm1c1N29vegVQz
1pvWls-SGNRNR25Wf132j4L0p2oxWmu5xLRFKm13Reo0mHcvT2xER1L5Q2w5ak4z03D0tN0lNvaVePsB2zgewQ0U3jSXZ2L0H0Hsz2dXeVnvHniqGxqfjdeT0fDf1Mz2CzNtVsakd1VraehJ4W
GtHk1pTULjweklqr08t0dA121l283RwcrjB8rbNaVkk4Y0Leh1Rj1xcFerdr09fiw1ZGF0Ymtle161kFRRUJS6SpCn99z3dnzRouHdhuX4u0lze0R1z54dudjLzQ4YkR3dndecE5Z
1pnQf05DR32kFZsktvwklodm0Q0VjR29H0Hdi1lQURC0b0na3fa21hOKcvQj3RXd1z11Kw1asUrxVRCQuV1TUJFRURia296SDBFVDvkyXdkdUfx0d1lC0eX81ljoREFUQV9LRVklc1leHbpcmf0aw9u1joxJ4Mmj00D1f0Q==
kYRszVzVMDesy0x5f1CbtZQozNkW11zNfZSwk10cnZtMs4ZERTR9nWlphTrz6ZTQ91wiwidmYcy2lVbi16j1j1lC0eX81ljoREFUQV9LRVklc1leHbpcmf0aw9u1joxJ4Mmj00D1f0Q==
.dkr.ecr.us-east-2.amazonaws.com

(base) C:\Users\saipr\Desktop\Final Project\chapter-08-serverless>$(aws ecr get-login --no-include-email)
$' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\saipr\Desktop\Final Project\chapter-08-serverless>aws ecr get-login --no-include-email
docker login -u AWS -p eyJwYXlsb2FkIjoiSURzQUVHTWxISng0NHdJNmhtCVgCwLdLSUxR0G1VmI1QyRnJBU2llicD1vekd20TBxRzRNMS9sZTixNTBmY0w5dG1zcXNQeEx20
5R8d591emg0Mzdt0Wx1ymEU2xJaT2FHJO2x0idhIpMy9jTHwaxBQ01xSwtDyKfVTNjsWd0tF4K02HeV1oYkr0UFBle1UrdndThQm1vB1ZE5tZQzNV2grm1iaT5sm2m7BqR0f5jBFmNDWFVEmphaj9FK
1pJ2ENrb1lMmfVNys3dGdXtnhUVK1zSwRb21VZQe1ZRT1dBNVJ0m2NzREVJ1UKRGy4Vge0edkYzhjcjd2WS9VOUROX16Smzpd2z14Lze1mNVTv14mPQVWhVpxRhoQz1NIAvkhZQm5V2B1n3pY081cGz1Vxd61cxzXnwV1InNfd2zuUm021uQ
3VYVmb0Su1YzJdeQsy1802X2m0EVx1d1H0nR02QzQs1pNkheQ8tYf1mzHNLw09hXfWmDF1zVnBrgt0a21DzQxbNvad1UtvexWhvSwSu1g2h2JchNv4BtRwnb1kbgHcm1c1N29vegVQz
zNeXg5yZhsca80e1m0kp2m1iaGovcTfG01psR0J0NEcncmc4Eue0H0dnd0BRkQTBl2zz22uDNbWQj01sZ1NvFf1R0d0tMbymhMeElyUmxtEz50Vh1y3R2u3m0u10251QxVcn1k0Q1nMrjt3t3hVgxleOpUd
1pblASQ0M4V4HpGmz20XBV8x1m5akm0131zVmlxUn0dtYzBxD0ZnRE1k0Wn1Qf1mzHNLw09hXfWmDF1zVnBrgt0a21DzQxbNvad1UtvexWhvSwSu1g2h2JchNv4BtRwnb1kbgHcm1c1N29vegVQz
1pvWls-SGNRNR25Wf132j4L0p2oxWmu5xLRFKm13Reo0mHcvT2xER1L5Q2w5ak4z03D0tN0lNvaVePsB2zgewQ0U3jSXZ2L0H0Hsz2dXeVnvHniqGxqfjdeT0fDf1Mz2CzNtVsakd1VraehJ4W
GtHk1pTULjweklqr08t0dA121l283RwcrjB8rbNaVkk4Y0Leh1Rj1xcFerdr09fiw1ZGF0Ymtle161kFRRUJS6SpCn99z3dnzRouHdhuX4u0lze0R1z54dudjLzQ4YkR3dndecE5Z
1pnQf05DR32kFZsktvwklodm0Q0VjR29H0Hdi1lQURC0b0na3fa21hOKcvQj3RXd1z11Kw1asUrxVRCQuV1TUJFRURia296SDBFVDvkyXdkdUfx0d1lC0eX81ljoREFUQV9LRVklc1leHbpcmf0aw9u1joxJ4Mmj00D1f0Q==
kYRszVzVMDesy0x5f1CbtZQozNkW11zNfZSwk10cnZtMs4ZERTR9nWlphTrz6ZTQ91wiwidmYcy2lVbi16j1j1lC0eX81ljoREFUQV9LRVklc1leHbpcmf0aw9u1joxJ4Mmj00D1f0Q==
.dkr.ecr.us-east-2.amazonaws.com

(base) C:\Users\saipr\Desktop\Final Project\chapter-08-serverless>
```

<https://813985511592.dkr.ecr.us-east-2.amazonaws.com>



813985511592.DKR.ECR.us-east-2.amazon.com/clothing-tflite-images:clothing-model-xception-v4-001

```

Anaconda Prompt (anaconda)
3ph1ckd0RnJuV0Z1dmY4dJzuvGozWDErZ0FqdWhFe1ZDd0RsVHpDRmhkc|Rtz2l1sE11N0ztz10rVgplswfpa2ptdFFy52doRmtXY1phvzFksuNSMmzT0FzbktxzjdTu1twTkdMj3BxVEzyMz1pSNV1U2t2dnZtC2h0bUN6V
^dm1M1Z0TwtzW9tBfnL0jzW956d2Craw76NUrmKkY1zzc2a1VYK3UTV1vd0xTe1VTUtrvzNLw0dzTDj0s1hVnRLV4V4M1t1U3F10DNgSoc2RDZKNExHWJKTklRz2B20TkyCTN1VXzjRVfSS1ce1kneTE1sOr1m0h8d
TN1bjIbfccfzSzW1J5nBEV1cd1RKt1zWsu1tUetVcwpd2xXbhTUh1kYop3V1Y7d6cnu47TU2eH73UzhneTMhUF1fNFzEbY0SXNRTNnsfhaulVkrXxyTcs3M1pkRv23mXZYYktrUjRGv5sWlRhWlOrSFauJnzb32YS
TN2ZzduMuRJMFO3119zKE52ewpQejbsRtdo2c3t1VxVtEydtB8dzM1wfVzSXh1n2j1e21M301tndGhrc2pDYUOvFeyRwz2wGZL2V15NwdpTQURVST20V1pjtDBsW445G95ZGVXdnZ1Wmd4UDNBQ1srRxhgZuWdYTj212Eke8N
HZXZf2TzVmnhUC900EVEdHBHXTUVQRTB0emlrcxVqRfFzTzQSMmhaaeFavpV32NMf1j112Ru2UTxhTA4bnltVxLu1ed5b29QZFmRkNEmmh5d1hvTWh2b3VzXQ03MDF1azBMSFByaHC1M1psM19wcvJFQGMNHJV0w1p1uhfn5
GtvBzRUDk5dTM0yWxoSNHMRHc1Ym5l5Wxu1d1z1e1b1elB2e0gtYm1ERDhzK7dXNz1ybEdtG15z6VVV10t9tiw1ZGF0Ymtle1G1kFRRUJBSpCNy99z3dnzROLhhdX4U01zeneR1z54dldj1zQ4YkR3dndEcE5ZV
1pnQJFBSDR32kfZSktrvk1lodm100vejR29H0HD1lU1lCQRcbnja3foa2lHOXc1QJ3Rxd121KwUlaSUXVXRQuV1TUfRURBGT1V0ppQ0n8xQm1zej01Q1UICRU1BNM2VXL2Fcc1NBV05S0xMU9CnNxt01zbnVR253Z
9940TcxRn2oVfhod0fml1ZY2o223pz1TQ1UKRt1nBnQvN1A2NEXj0HhCudQNTB3NzQ91iwidnvC2lvB1i1j1Cj0eX81j1j1oiREFQV9LRVki1Cj1leHbpcmf0aw9u1joxNjhMhjic@NT1sfQ== https://813985511592
.dkr.ecr.us-east-2.amazonaws.com

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>ACCOUNT=813985511592
'ACCOUNT' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>REGION=us-east-2
'REGION' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>REGISTRY=clothing-tflite-images
'REGISTRY' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>REGISTRY_URL=${ACCOUNT}.dkr.ecr.${REGION}.amazon.com/${REGISTRY}
'REGISTRY_URL' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>
(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>TAG=clothing-model-xception-v4-001
'TAG' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>REMOTE_URI=${PREFIX}: ${TAG}
'REMOTE_URI' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>clear
'clear' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
clothing-model     latest   ac66d438ab42   19 hours ago  1.08GB

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>
```

Tag this image with the URI that we have just created.

docker tag clothing-model:latest \${REMOTE\_URI}

```

Anaconda Prompt (anaconda)
#7 2.652 ERROR: tflite_runtime-2.2.0-cp37-cp37m-linux_x86_64.whl is not a supported wheel on this platform.
-----
executor failed running [/bin/sh -c pip3 install https://raw.githubusercontent.com/alexeygrigorev/serverless-deep-learning/master/tflite/tflite_runtime-2.2.0-cp37-cp37m
-linux_x86_64.whl --no-cache-dir]: exit code: 1

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker tag clothing-tflite-images:latest 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-
images:latest
Error response from daemon: No such image: clothing-tflite-images:latest

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images     latest   ac66d438ab42   20 hours ago  1.08GB
clothing-model      latest   ac66d438ab42   20 hours ago  1.08GB
cloker/getting-started          latest   eb9194091564   2 weeks ago  28.5MB

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker tag clothing-tflite-images:latest 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-model:1
atest
Error response from daemon: No such image: clothing-tflite-images:latest

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images     latest   ac66d438ab42   20 hours ago  1.08GB
clothing-model      latest   ac66d438ab42   20 hours ago  1.08GB
cloker/getting-started          latest   eb9194091564   2 weeks ago  28.5MB

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker tag clothing-model:latest 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest
atest

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>docker push 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest
The push refers to repository [813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images]
1b10afe01aea: Pushed
af25ae01aea: Pushed
d45259eca969: Pushed
40e799ea427f: Pushed
3bcc612abb5: Pushed
5411551195dd: Pushed
52375f4c3c68: Pushed
f6ae2f36d5d7: Pushed
d5a75698184d: Pushed
e839a65cd20b: Pushed
Rdfraab7ee286: Pushed
latest: digest: sha256:a3f5eb246922d5d65d1b65d6258e8478a44482f4e39c91115a6d0f1f37877f2a size: 2634

(base) C:\Users\sapir\Desktop\Final Project\chapter-08-serverless>
```

```
docker tag clothing-model:latest 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest
```

```
docker push 813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images:latest
```

The screenshot shows the Amazon ECR console with the URL [us-east-2.console.aws.amazon.com/ecr/repositories/private/813985511592/clothing-tflite-images?region=us-east-2](https://us-east-2.console.aws.amazon.com/ecr/repositories/private/813985511592/clothing-tflite-images?region=us-east-2). The left sidebar shows the navigation menu for Amazon Container Services, with 'Amazon ECR' and 'Images' selected. The main content area displays the 'clothing-tflite-images' repository. It shows one image entry:

Image tag	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
latest	November 29, 2021, 19:49:40 (UTC-06)	399.73	Copy URI	sha256:a3f5eb246922d5...	-	-

The screenshot shows the Amazon ECR console with the URL [us-east-2.console.aws.amazon.com/ecr/repositories?region=us-east-2](https://us-east-2.console.aws.amazon.com/ecr/repositories?region=us-east-2). The left sidebar shows the navigation menu for Amazon Container Services, with 'Amazon ECR' and 'Repositories' selected. The main content area displays the 'Private repositories' list. It shows one repository entry:

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
clothing-tflite-images	813985511592.dkr.ecr.us-east-2.amazonaws.com/clothing-tflite-images	November 29, 2021, 18:02:39 (UTC-06)	Disabled	Manual	AES-256	Inactive

## Creating ECR Container Image

The screenshot shows the 'Create function' wizard in the AWS Lambda console. The 'Container image' option is selected, highlighted with a blue border. The 'Basic information' section is visible, showing fields for 'Function name' (set to 'myFunctionName') and 'Container image URI' (set to 'clothing-tflite-images'). Other options like 'Author from scratch' and 'Use a blueprint' are also shown.

The screenshot shows the 'Select container image' dialog box. It displays the 'Amazon ECR image repository' field set to 'clothing-tflite-images'. Below it, the 'Images' section shows a single image entry: 'latest' with a digest value of 'sha256:a3f5eb246922d5d65d1b65d6258e8478a444 82f4e39c91115a6d0f1f37877f2a'. The 'Select image' button is visible at the bottom right of the dialog.

The screenshot shows the AWS Lambda console interface. At the top, there are several tabs and a search bar. Below the search bar, the main title is "Creating the function clothing-classification". Under this, the function name "clothing-classification" is displayed. On the left, there's a sidebar with "Function overview" and "Info" sections. The "Info" section shows a thumbnail of the function, a "Description" field (empty), a "Last modified" timestamp ("2 seconds ago"), and a "Function ARN" field containing "arn:aws:lambda:us-east-2:813985511592:function:clothing-classification". Below the "Info" section are tabs for "Image", "Test", "Monitor", "Configuration", "Aliases", and "Versions". The "Image" tab is currently selected. At the bottom of the page, there's a navigation bar with links like "Feedback", "English (US)", and "Cookie preferences".

## INVOKING THE FUNCTION WITH THE TEST EVENT – increase the timeout to 30 seconds

The screenshot shows the AWS Lambda console interface, specifically the "Test" tab. A green banner at the top says "Your changes have been saved.". The main area is titled "Test event" and contains instructions: "Invoke your function with a test event. Choose a template that matches the service that triggers your function, or enter your event document in JSON." There are two radio button options: "New event" (selected) and "Saved event". Below this is a "Template" dropdown set to "hello-world" and a "Name" input field containing "MyEventName". To the right of the input fields is a large text area showing a JSON template:

```
1 - [  
2 - {  
3 -   "url": "http://bit.ly/mlbookcamp-pants"  
4 - }]
```

At the bottom of the page, there's a navigation bar with links like "Feedback", "English (US)", and "Cookie preferences".

**Test Event Settings**

Basic settings [Info](#)

Description - optional

Memory [Info](#)  
Your function is allocated CPU proportional to the memory configured.

1024 MB  
Set memory to between 128 MB and 10240 MB

Timeout  
1 min 0 sec

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role

Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

service-role/clothing-classification-role-8dnwkiwn

[View the clothing-classification-role-8dnwkiwn role on the IAM console](#).

Feedback English (US) ▾ © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

clothing-classification - Lamb [Edit basic settings](#) ['sed' is not recognized as an internal or external command, operable program or batch file](#) [How do I push a Docker image to Amazon Lambda?](#) [How to push Docker images to Lambda](#) [Troubleshooting errors with Lambda](#) [+ New tab](#)

us-east-2.console.aws.amazon.com/lambda/home?region=us-east-2#/functions/clothing-classification?newFunction=true&tab=testing

Feedback English (US) ▾ © 2021, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
{ "dress": -1.8682897090911865, "hat": -4.761244773864746, "longsleeve": -2.3169829845428467, "outwear": -1.0625699758529663, "pants": 9.887154579162598, "shirt": -2.8124303817749023, "shoes": -3.666283130645752, "shorts": 3.2003610134124756, "skirt": -2.602339506149292, }
```

Summary

Code SHA-256	Request ID
a3f5eb246922d5d5d1b65d6258e8478a44482f4e39c91115a6d0f1f37877f2a	b5e521b0-c4e8-4747-b436-824907993306
Init duration	Duration
5035.61 ms	2607.78 ms
Billed duration	Resources configured
7644 ms	1024 MB

## **Future Work**

- Understanding different methods of deploying and serving models in the cloud.
- Serving Keras and TensorFlow models with TensorFlow-Serving
- Deploying TensorFlow-Serving to Kubernetes